CLAIMS

What is claimed is:

1. A method, comprising:

buffering packet header and payload data corresponding to a plurality of inbound transmission control protocol (TCP) packets received at a destination machine in an inbound buffer; and

performing TCP input processing of the packet header and payload data that is buffered in the inbound buffer via a multi-threaded hardware engine, wherein multiple hardware-arbitrated threads are concurrently executed by the multi-threaded hardware engine to process the plurality of inbound TCP packets.

2. The method of claim 1, further comprising arbitrating thread processing via a hardware-based scheduler.

3. The method of claim 2, wherein arbitrating thread processing comprises performing at least one of thread suspension, thread scheduling, thread synchronizing, saving thread state and restoring thread state.

4. The method of claim 1, wherein the multi-threaded hardware engine comprises a dedicated TCP offload engine (TOE).

5. The method of claim 1, further comprising concurrently transferring payload data buffered in the inbound buffer to host memory while performing TCP input processing via the multi-threaded hardware engine.

6. The method of claim 5, further comprising performing a direct memory access (DMA) transfer to concurrently transfer the payload data to host memory.

7. The method of claim 5, further comprising pre-posting memory locations in the host memory to which payload data is to be transferred.

8. The method of claim 1, further comprising:

determining an existence of a TCP connection;

generating TCP connection context data corresponding to the TCP connection; and

storing the TCP connection context data in host memory.

9. The method of claim 8, further comprising maintaining a cache in which selected TCP connection context data is cached.

10. The method of claim 9, further comprising:

retrieving the TCP connection context data for a given packet from one of host memory or the cache;

loading the TCP connection context data into a working register; and

processing the TCP connection context data via the multi-threaded hardware engine to perform TCP input processing.

11. The method of claim 10, further comprising:

performing a hash-based lookup against the cache to determine if the TCP connection context data for the given packet is present in the cache; and

loading the TCP connection context data from the cache into the working register if the hash-based lookup results in a cache hit, otherwise copying the TCP

connection context data from host memory into the cache prior to loading the TCP connection data into the working register.

12. A method comprising:

generating transmission control protocol (TCP) connection context data corresponding to a TCP connection employed to transmit payload data stored in host memory from a host machine to a destination machine; and

performing TCP output processing of the payload data stored in memory via a multi-threaded hardware engine running on the host machine, wherein multiple hardware-arbitrated threads are concurrently executed by the engine to generate a plurality of outbound TCP packets containing the payload data, each outbound TCP packet including a header containing TCP connection data corresponding to the TCP connection context data.

13. The method of claim 12, further comprising arbitrating thread processing via a hardware-based scheduler.

14. The method of claim 13, wherein arbitrating thread processing comprises performing at least one of thread suspension, thread scheduling, thread synchronizing, saving thread state and restoring thread state.

15. The method of claim 12, wherein the multi-threaded hardware engine comprises a dedicated TCP offload engine (TOE).

16. The method of claim 12, further comprising concurrently transferring data comprising outbound TCP packets from host memory to a network interface

controller (NIC) while performing TCP output processing via the multi-threaded hardware engine.

17. The method of claim 16, further comprising performing a direct memory access (DMA) transfer to concurrently transfer the data comprising outbound TCP packets from the host memory to the NIC.

18. The method of claim 17, further comprising maintaining a DMA transmit queue containing information defining how DMA transfers are queued.

19. The method of claim 12, further comprising maintaining a cache in which selected TCP connection context data is cached.

20. The method of claim 19, further comprising:

retrieving the TCP connection context data for a given portion of payload data from one of host memory or the cache;

loading the TCP connection context data into a working register; and

processing the TCP connection context data via the multi-threaded hardware engine to perform TCP output processing.

21. The method of claim 20, further comprising:

performing a hash-based lookup against the cache to determine if the TCP connection context data for the given portion of payload data is present in the cache; and

loading the TCP connection context data from the cache into the working register if the hash-based lookup results in a cache hit, otherwise copying the TCP

connection context data from host memory into the cache prior to loading the TCP connection data into the working register.

22. An integrated circuit, comprising:

a multi-threaded transmission control protocol (TCP) offload engine (TOE), including:

a processing engine;

a scheduler, communicatively coupled to the processing engine;

a host memory interface, communicatively coupled to the processing engine; and

a network interface controller (NIC) interface; communicatively coupled to the processing engine; and

a direct memory access (DMA) controller, communicatively coupled to the NIC interface and the host memory interface.

23. The integrated circuit of claim 22, further comprising a cache communicatively coupled to the processing engine and the host memory interface.

24. The integrated circuit of claim 22, further comprising a host interface communicatively coupled to the processing engine.

25. The integrated circuit of claim 22, wherein the processing engine comprises:

a pipelined arithmetic logic unit (ALU);

a working register, communicatively coupled to the pipelined ALU;

an instruction cache to store instructions executable by the pipelined ALU; and

an instruction register, communicatively coupled between the instruction cache and the pipelined ALU.

26. The integrated circuit of claim 25, wherein the processing engine further includes a thread cache, communicatively coupled to the working register.

27. The integrated circuit of claim 22, wherein the integrated circuit comprises a memory controller hub (MCH) in a platform chipset.

28. A system, comprising:

at least one processor, communicatively coupled to a frontside bus;

host memory communicatively coupled to a memory bus; and

a memory controller hub (MCH) communicatively coupled to the at least one processor via the frontside bus and the host memory via the memory bus, the MCH embodied as an integrated circuit comprising:

a multi-threaded transmission control protocol (TCP) offload engine (TOE), including:

a processing engine;

a scheduler, communicatively coupled to the processing engine;

a host memory interface, communicatively coupled to the processing engine and the memory bus;

a host interface, communicatively coupled to the processing engine and the frontside bus;

a network interface controller (NIC) interface; communicatively coupled to the processing engine; and

a direct memory access (DMA) controller, communicatively coupled to the NIC interface and the host memory interface.

29. The system of claim 28, further comprising a network interface controller (NIC), communicatively coupled to the NIC interface via one of a PCI (peripheral component interconnect) or PCI-X (PCI Express) bus.

30. The system of claim 28, wherein the MCH further includes a cache communicatively coupled to the processing engine and the host memory interface.